A Tale of Two Timelines

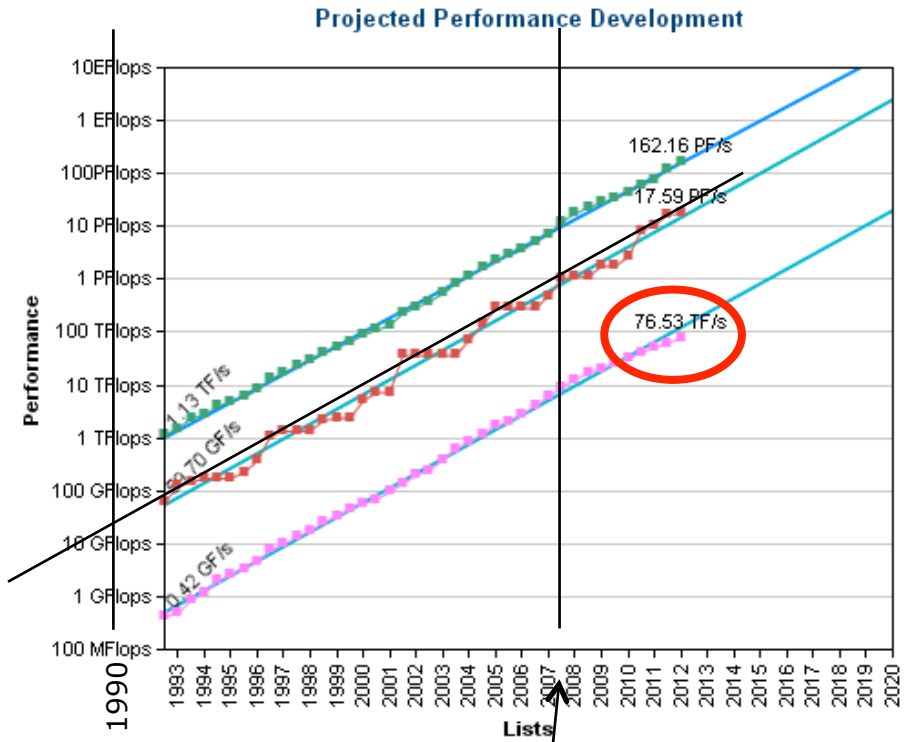
William Gropp
www.cs.illinois.edu/~wgropp

Introduction

- Computer science has a least two timescales:
  - Very fast: esp. the increase in hardware capability
  - Slow or step changes: everything else
- Sustaining progress requires recognizing the difference between these

Amazing Increase in Computing Power

- Exponential increase in performance for several *decades*
- Five (!) orders of magnitude while I was in MCS
- But not everything has changed that fast…



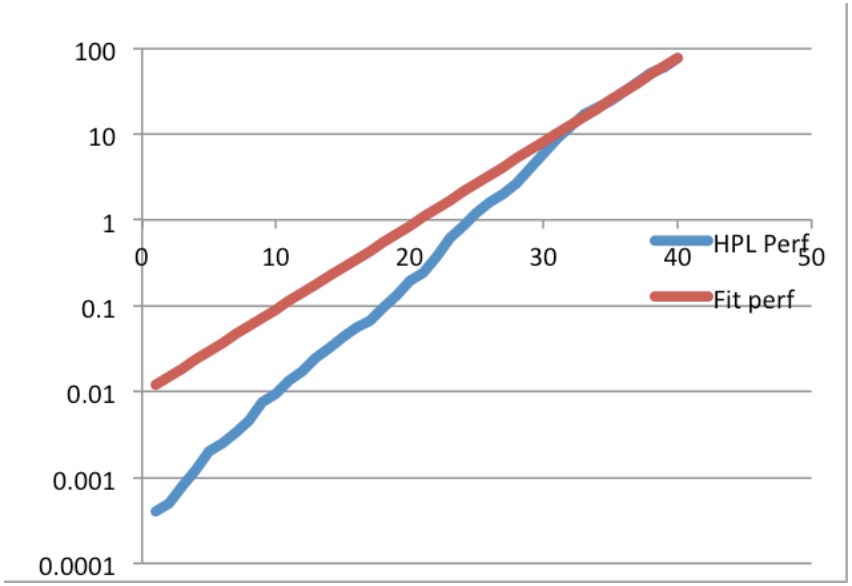
I Start at MCS

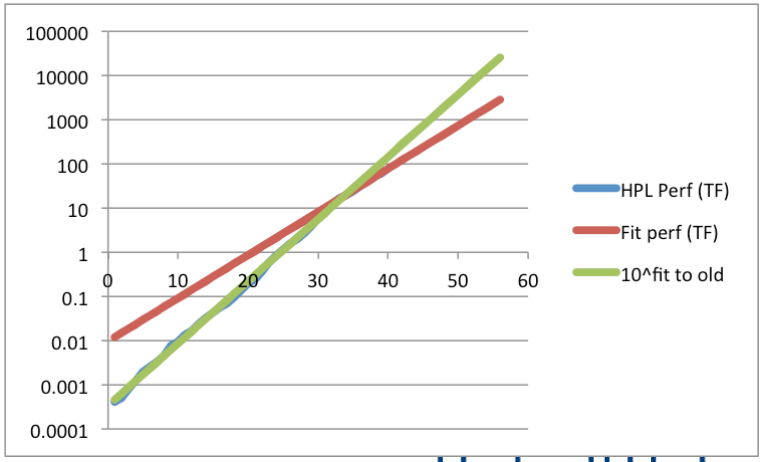
I Leave MCS

PARALLEL@ILLINOIS

That "kink" in #500 is Real

- Extrapolation of recent data gives ~1PF HPL in 2018 on the #500 system
- Extrapolation of older data gives ~1PF in 2015, ~7PF in 2018
- The #500 *may* be a better predictor of trends

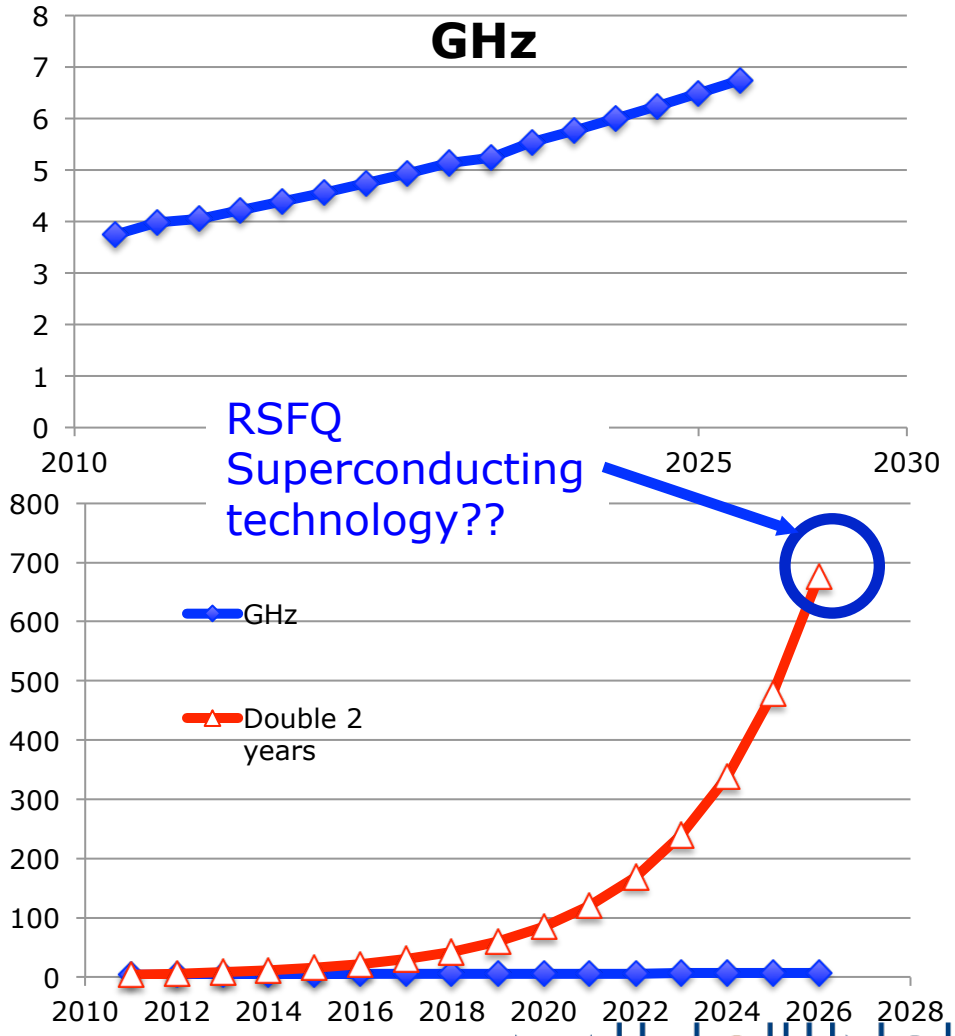




PARALLEL@ILLINOIS

Frequency Scaling is Over

- New (prediction): Increase 4% per year (ITRS 2012 Roadmap)
- Old: Double every 2 years
- The change (loss) is enormous
- Extrapolations are just as dangerous as we tell our students



PARALLEL@ILLINOIS

Everything Else Changes Slowly

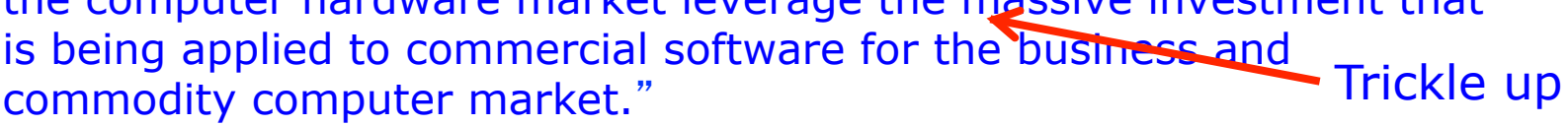
- Programming, libraries
- Standards, software, languages
  - Ken Kennedy said it takes at least 10 years for a new programming language to "take"
  - MPI and MPICH illustrate both (see later)
- Somewhere in the middle
  - Are Applications here? What do you think?
- "Punctuated Equilibrium" may be a better model
  - Combined with slow change
  - Can argue that accelerators are another step change in hardware (look at the *top* of the top500)
- To predict the future it is useful to look at the past…

Quotes from "System Software and Tools for High Performance Computing Environments" (1993)

- "The strongest desire expressed by these users was simply to satisfy the urgent need to get applications codes running on parallel machines as quickly as possible"
- In a list of enabling technologies for mathematical software, "Parallel prefix for arbitrary user-defined associative operations should be supported. Conflicts between system and library (e.g., in message types) should be automatically avoided."
  - Note that MPI-1 provided both
- Immediate Goals for Computing Environments:
  - Parallel computer support environment
  - Standards for same
  - Standard for parallel I/O
  - Standard for message passing on distributed memory machines
- "**The single greatest hindrance** to significant penetration of MPP technology in scientific computing is **the absence of common programming interfaces** across various parallel computing systems"

Quotes from "Enabling Technologies for Petaflops Computing" (1995)

- "**The software for the current generation of 100 GF machines is not adequate to be scaled to a TF…**"
- "The Petaflops computer is achievable at reasonable cost with technology available in about 20 years [2014]."
  - (estimated clock speed in 2004 — 700MHz)*
- "Software technology for MPP's must evolve new ways to design software that is portable across a wide variety of computer architectures. Only then can the small but important MPP sector of the computer hardware market leverage the massive investment that is being applied to commercial software for the business and commodity computer market."
- "To address the inadequate state of software productivity, there is a need to develop language systems able to integrate software components that use different paradigms and language dialects."
- (9 overlapping programming models, including shared memory, message passing, data parallel, distributed shared memory, functional programming, O-O programming, and evolution of existing languages)



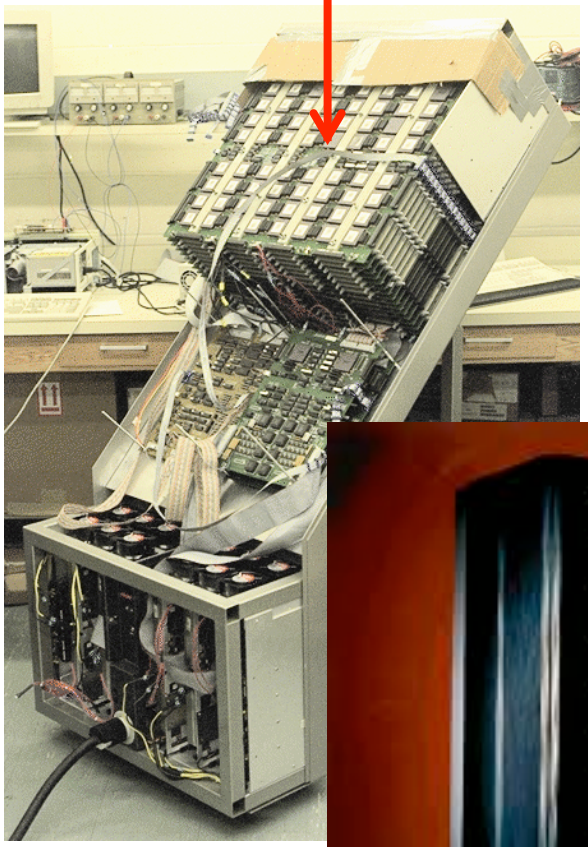
Why This Matters

- Performance gains from hardware are slowing
  - Some features, such as frequency scaling, ended years ago
  - We need to change intuition about hardware performance and impact on algorithms and software
- Expectations of rapid change diverts attention from the need to sustain development in software and algorithms
  - Change is a step – but the step only succeeds if it is nurtured

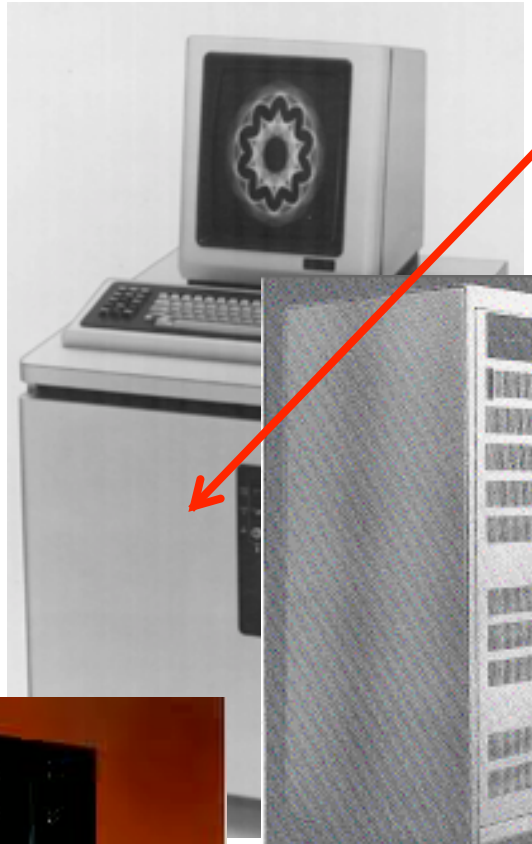
30 years of Parallel Computing at MCS

- Me: 17 of those, until 5 years ago
  - ♦ Many major transitions:
    - Shared memory (Encore Multimax) to mixed (BBN TC2000) to distributed memory (everything since) to mixed (SMP+RDMA)
  - ♦ "Scalable" goes from 100x to 1,000,000x
  - ♦ Software makes big strides in productivity for computational scientists
    - Numerical libraries (PETSc – '90 – '96)
    - Parallel Computing (MPI/MPICH – '92 – '07 – now)

Do you recognize this machine?



"My" First Computer (50KFLOPS!)



Research in Numerical Analysis

- Problem: Performing research into parallel *domain decomposition* algorithms
  - Divide domain into parts, solve on parts, put back together
  - May want to recurse (solve by applying domain decomposition)
- Most numerical libraries of the time unusable
  - Global state: Can't nest library calls; some have high overhead for initialization
  - No routines to solve problems – only routines to apply a specific algorithm
  - Often "unnatural" data structures (designed for algorithm, not problem)
  - No parallelism

Solution: A New Way of Looking at Numerical Libraries

- No global state – encapsulate data *and functions* needed by routine
  - ♦ Enables algorithms that are the composition of others
- Organized by *operation* not algorithm
  - ♦ Which algorithm *and data structure* is part of the state
  - ♦ Enables polyalgorithms where algorithm choice is data dependent
  - ♦ Parallelism is (mostly) hidden

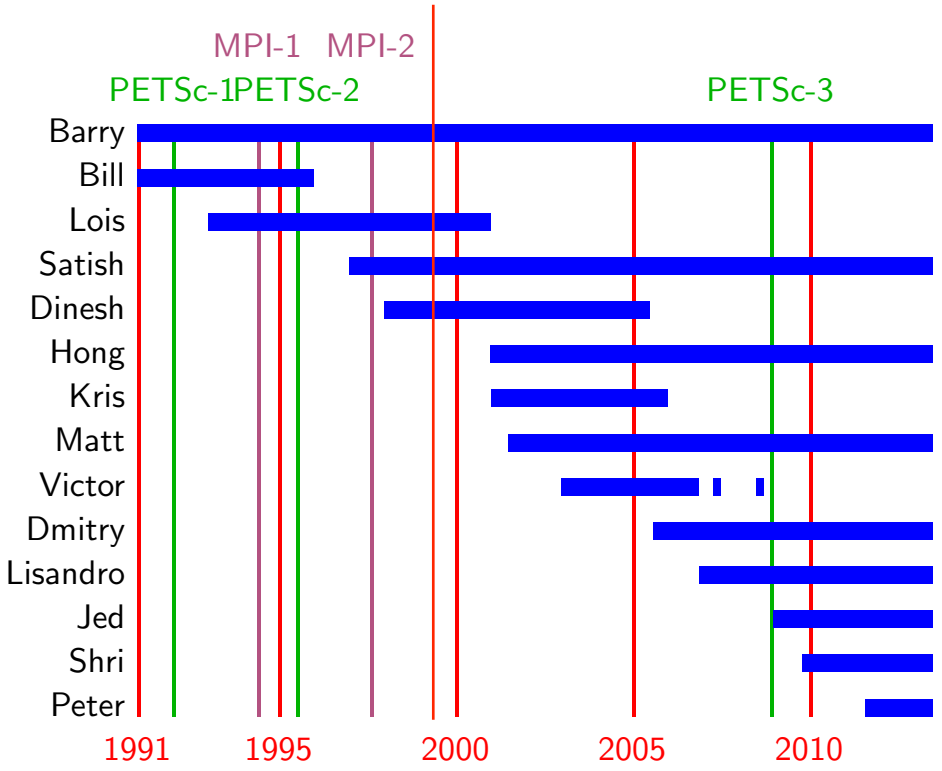
PETSc

- PETSc was originally a portable library for solving linear and nonlinear systems of equations in parallel
  - PETSc was designed to provide a library for experimentation in domain decomposition algorithms
- PETSc now best described as a suite of data structures and routines for the scalable parallel solution of scientific applications modeled by partial differential equations
- Initial version allow Barry Smith and me to conduct our research (fast change)
- Extending PETSc to meet needs of other applications, researchers required sustained effort…

Changing Numerical Libraries

- Two very distinct timescales
  - Fast: New way of looking at organization
  - Slow: Work of implementation, tuning, extension
- Requires sustained effort to provide end-to-end support, extend to new application needs

First Gordon Bell Prize



"Why we couldn't use numerical libraries for PETSc," Proceedings of the IFIP TC2/WG2.5 Working Conference on the Quality of Numerical Software, Assessment and Enhancement, 1997.

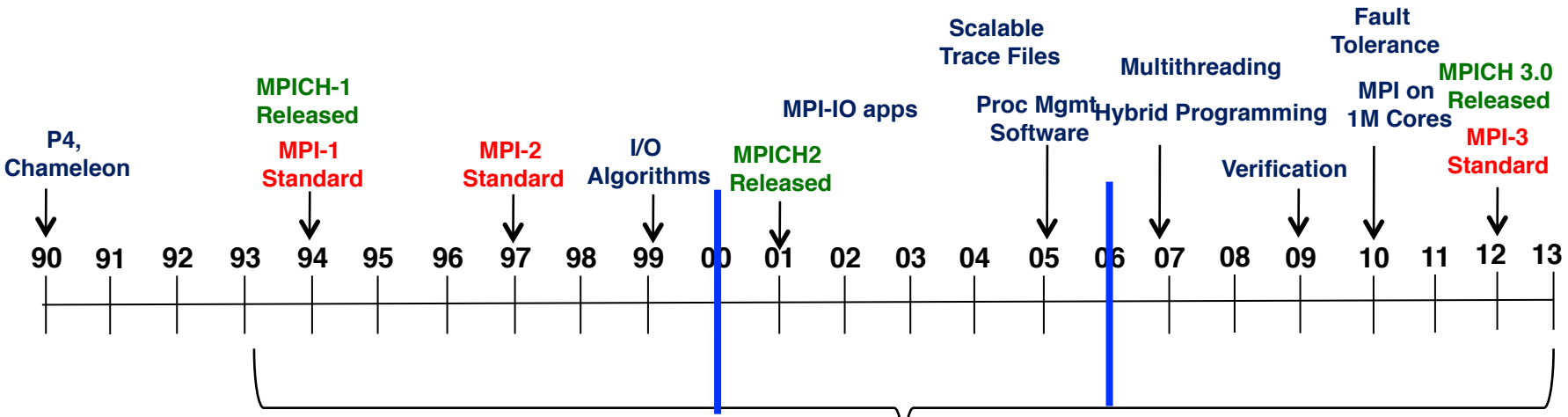
Making it Safe for Parallel Software

- 1993: "**The single greatest hindrance** to significant penetration of MPP technology in scientific computing is **the absence of common programming interfaces** across various parallel computing systems"
- How do we overcome this problem?

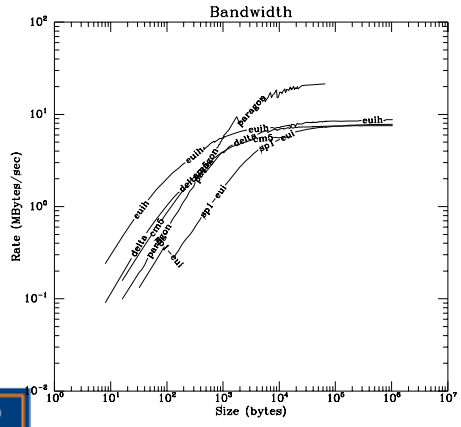
MPI and MPICH

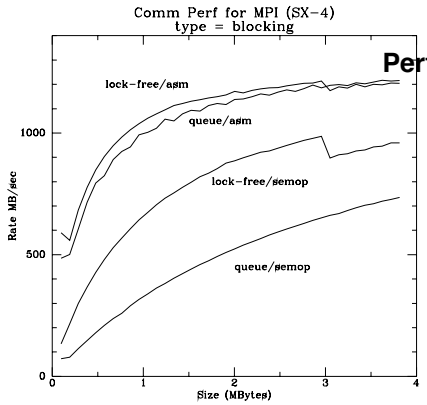
- Develop a *standard* that embodies the best ideas from the community
  - Standard makes portability possible
  - Community involvement improves completeness, design
  - ANL/MCS's experience in portability (p4) and performance (Chameleon) contributed to MPI's design
- Develop an implementation *designed* to provide performance and exploit special hardware
  - Not *just a* reference implementation
  - Solve performance and productivity issues with new algorithms and implementation ideas
  - Stay connected to applications (DOE Mission, others) and vendors (IBM, Cray, NEC, SGI, others)
  - But took a sustained effort…
- *Learning from the Success of MPI*, HiPC 2001

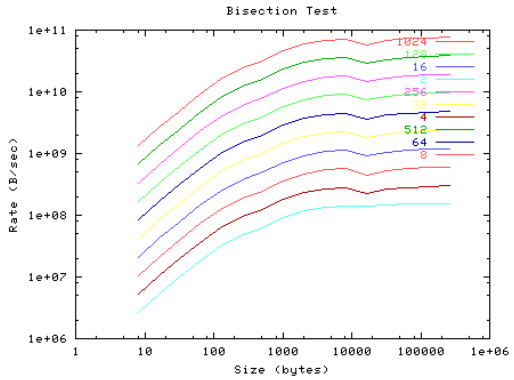
MPI and MPICH Timeline



Performance research

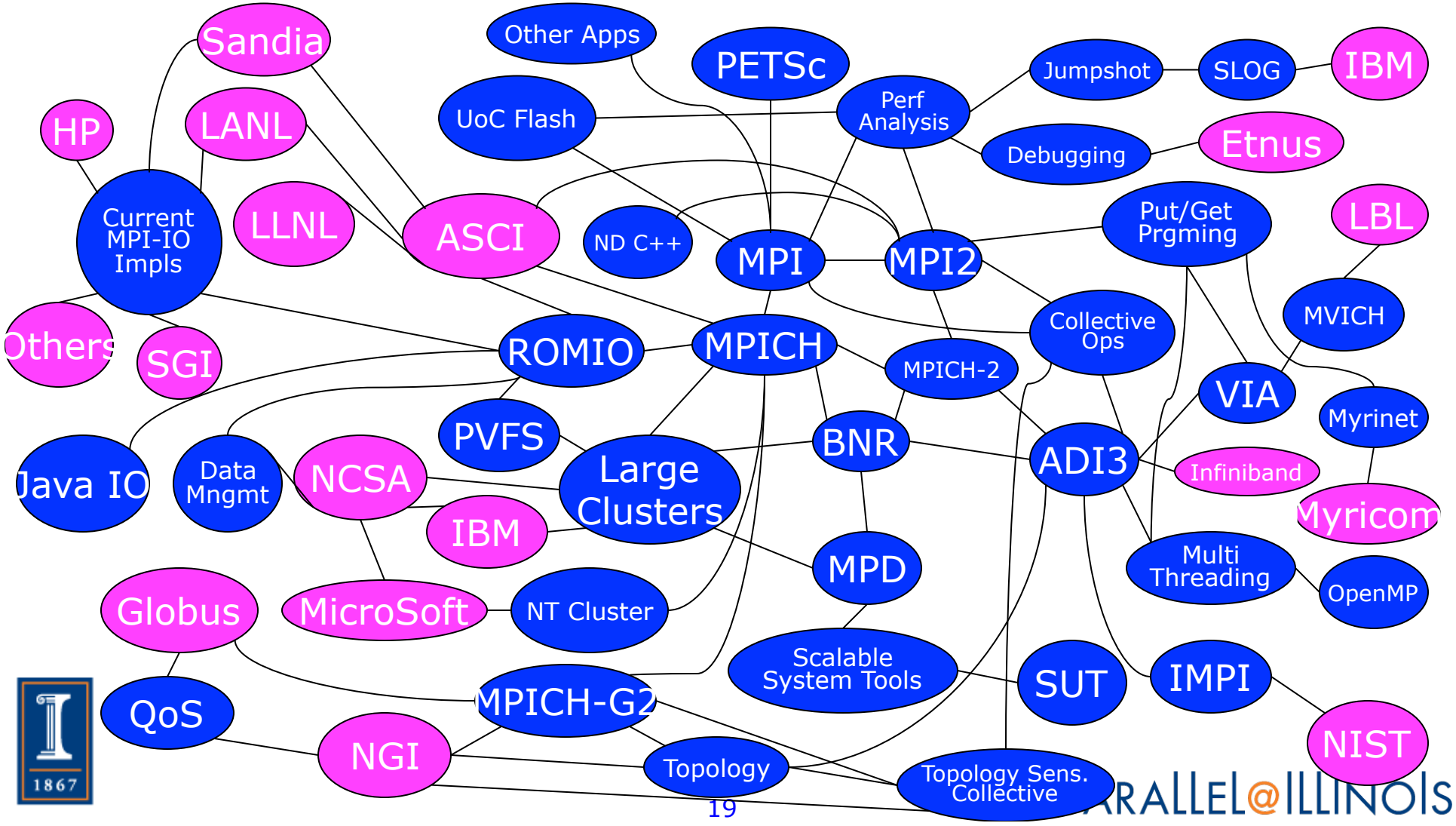




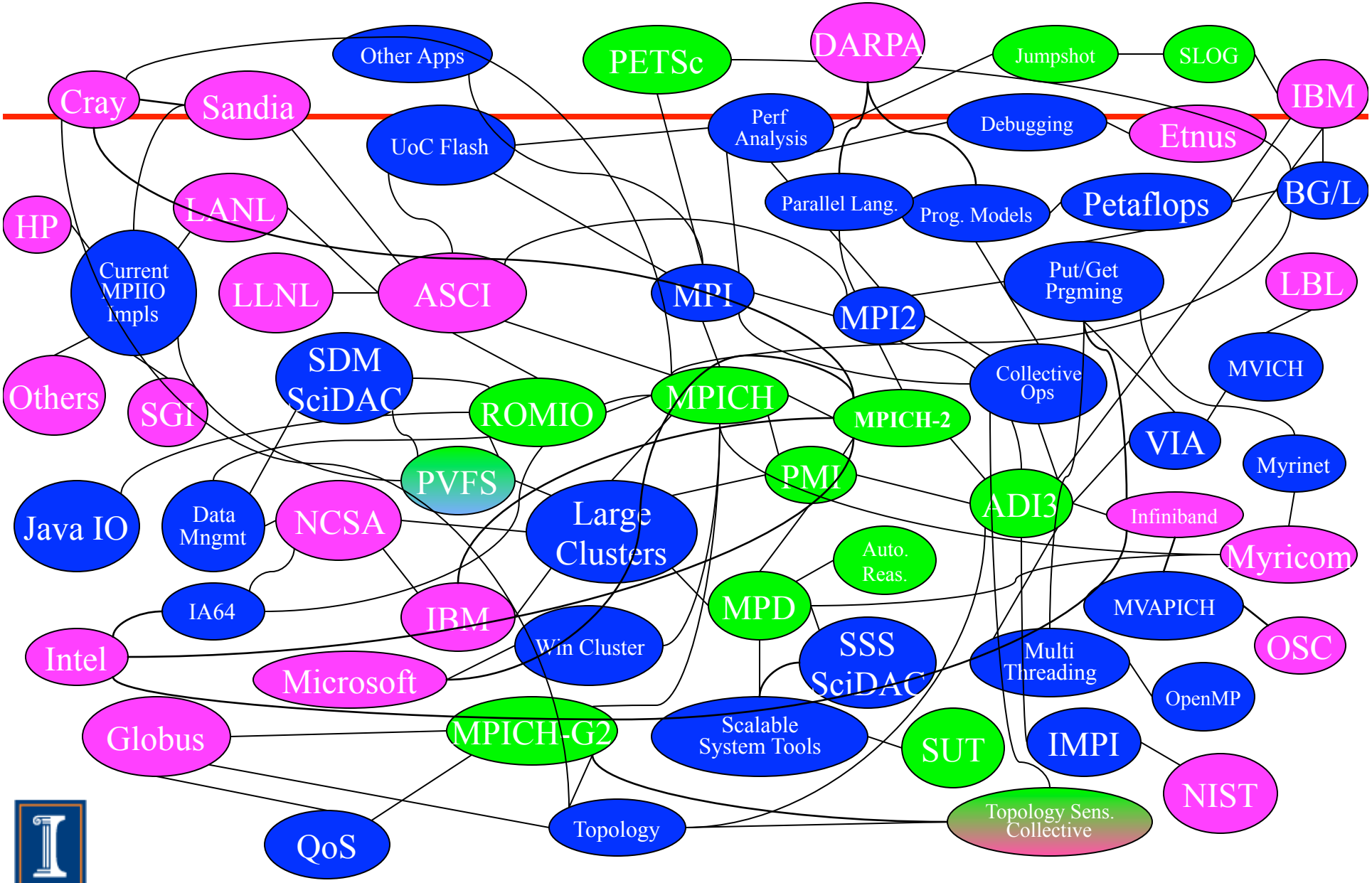


PARALLEL@ILLINOIS

MPICH and the World



MPICH2 and the World



PARALLEL@ILLINOIS

Summary

- Implementation takes time
  - More than just "writing code"
  - Involves research into methods, understanding of application needs
  - Feedback essential in making progress
- Step changes *are po*ssible
  - But they don't succeed immediately
  - Q: When was the GPU introduced by NVIDIA?
    - Founded 1993, GeFORCE in 1999, CUDA 2006
- Progress requires an environment that supports the work of making a revolution succeed
  - MCS and DOE provided this environment and enabled the parallel computing revolution